UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/779,453	02/09/2001	Christopher C. Tanner	2198.0070002	6696

26111 7590 12/16/2005

STERNE, KESSLER, GOLDSTEIN & FOX PLLC
1100 NEW YORK AVENUE, N.W.
WASHINGTON, DC 20005

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 12/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/779,453	**Applicant(s)** TANNER ET AL.	
	Examiner Michael J. Yigdall	**Art Unit** 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 September 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *20-23* is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to Applicant's submission filed on September 19, 2005. Claims 1-23 are pending.

Response to Amendment

2. The rejection of claims 20-23 under 35 U.S.C. 112, second paragraph, is withdrawn in view of Applicant's amendment.

Response to Arguments

3. Applicant's arguments with respect to the limitation(s) added to claims 1, 3, 5, 8, 14, 16 and 18 have been considered but are moot in view of the new ground(s) of rejection.

4. Applicant's other arguments have been considered but they are not persuasive.

Applicant contends that Pavan does not teach "allow[ing] the execution of an application to be modified or changed while the application itself is executing," and contends that Pavan, as cited, "merely refers to the process for translating a user program to a system-level program" (Applicant's remarks, page 12, first complete paragraph). However, a translation technology is not inherently a compile-time technology, as Applicant would suggest (Applicant's remarks, page 12, top paragraph). On the contrary, the translation taught by Pavan is performed at runtime when the application is submitted for execution (see, for example, column 12, lines 1-13). The program is modified or changed dynamically at runtime while the program is executing (see, for example, column 8, line 51 to column 9, line 9).

Applicant alleges generally that "the Examiner has not provided a suggestion or motivation to combine two or more of the cited documents" (Applicant's remarks, page 12, last paragraph). However, Applicant is respectfully reminded that such a suggestion or motivation has been provided in earlier Office actions and is again provided below, with repeated citations for further clarity.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-8, 10, 11 and 16-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,578,197 to Peercy et al. (art of record, "Peercy") in view of U.S. Patent No. 6,502,238 to Pavan et al. (art of record, "Pavan") in view of U.S. Patent No. 6,823,299 to Contreras et al. (art of record, "Contreras").

With respect to claim 1 (currently amended), Peercy discloses a method for supporting development of content independent of a run-time platform (see, for example, the abstract and FIG. 1).

Although Peercy discloses using procedures and functions to define graphics content (see, for example, column 6, lines 10-20), Peercy does not expressly disclose the step of:

(a) storing processing blocks that define content.

However, Pavan discloses storing processing blocks, accessible to the user, that define the content of a program (see, for example, column 4, lines 28-37). The processing blocks encapsulate functions performed by hardware and functions that are otherwise hidden from the user (see, for example, column 4, lines 9-18).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy with processing blocks, such as taught by Pavan, so as to encapsulate and provide access to functions performed by hardware and functions that are otherwise hidden from the user (see, for example, Pavan, column 4, lines 9-18).

Although Peercy discloses storing a tree or application graph that represents a graphics computation (see, for example, step 208 in FIG. 2, FIG. 3 and column 4, lines 14-16), and further discloses traversing the tree in a graphical application platform (see, for example, step 210 in FIG. 2) to execute the graphics computation (see, for example, steps 212 and 214 in FIG. 2), Peercy does not expressly disclose the step of:

(b) storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks;

wherein the application graph can be traversed by a graphical application platform at run time to execute appropriate processing blocks on a run time platform.

However, Pavan further discloses storing an application graph (see, for example, user program 400 in FIG. 4) that represents the identity of the stored processing blocks and the data connectivity between the blocks (see, for example, column 5, lines 8-14, and column 6, lines 18-26). Pavan further discloses that the application graph is traversed and translated at runtime to execute appropriate processing blocks on a runtime platform (see, for example, column 8, line 51 to column 9, line 9). The traversal is performed at runtime when the application is submitted for execution (see, for example, column 12, lines 1-13). Program fragments are automatically and transparently created and distributed for execution over a network based on the application graph (see, for example, column 11, lines 50-63).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy with an application graph, such as taught by Pavan, so as to automatically and transparently create and execute a distributed program for the graphics computation of Peercy (see, for example, Pavan, column 11, lines 50-63).

Although Pavan discloses that the application graph can be modified at runtime (see, for example, column 8, lines 60-67, which shows inserting blocks, and column 11, lines 2-12 and 20-32, which shows modifying the connections between the blocks) when the application is submitted for execution (see, for example, column 12, lines 1-13), Peercy in view of Pavan does not expressly disclose the limitation:

wherein an execution of at least one of the appropriate processing blocks can cause the application graph to be modified by the graphical application platform.

However, Contreras discloses an application graph (see, for example, FIG. 3) and an engine that traverses the application graph at runtime (see, for example, column 5, lines 13-16). The application graph includes nodes or processing blocks (see, for example, column 4, lines 27-36) and edges that represent data flow connections (see, for example, column 4, lines 37-44). Contreras further discloses a database or dictionary of nodes (see, for example, column 4, lines 9-11). Contreras expressly discloses modifying the connections among nodes (see, for example, column 5, lines 25-27) and adding nodes to the application graph dynamically at runtime (see, for example, column 5, lines 37-39). The execution of a node or processing block can cause such changes to the application graph, so as to respond to events dynamically without user intervention (see, for example, column 4, lines 45-52 and column 5, lines 55-59).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy and Pavan with the execution features taught by Contreras, so as to respond to events dynamically without user intervention (see, for example, Contreras, column 4, lines 45-52).

With respect to claim 2 (original), Peercy further discloses the limitation wherein the content comprises game content (see, for example, column 1, lines 16-23, which shows developing graphics applications for the entertainment industry, and column 11, lines 1-2, which further shows using video game cartridges).

With respect to claim 3 (currently amended), Peercy discloses a method for supporting development of content independent of multiple hardware platforms (see, for example, the abstract and FIG. 1).

Although Peercy discloses using procedures and functions to define graphics content independent of multiple hardware platforms (see, for example, column 6, lines 10-20), Peercy does not expressly disclose the step of:

(a) storing processing blocks that define content independent of multiple hardware platforms.

However, Pavan discloses storing processing blocks, accessible to the user, that define the content of a program (see, for example, column 4, lines 28-37) independent of multiple hardware platforms (see, for example, column 5, lines 25-32). The processing blocks encapsulate functions performed by hardware and functions that are otherwise hidden from the user (see, for example, column 4, lines 9-18).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy with processing blocks, such as taught by Pavan, so as to encapsulate and provide access to functions performed by hardware and functions that are otherwise hidden from the user (see, for example, Pavan, column 4, lines 9-18).

Peercy further discloses the step of:

(b) selecting a target hardware platform from multiple hardware platforms (see, for example, step 140 in FIG. 1 and column 6, lines 37-39, which shows targeting or selecting a hardware platform).

Although Peercy discloses storing a tree or application graph that represents a graphics computation (see, for example, step 208 in FIG. 2, FIG. 3 and column 4, lines 14-16), and further discloses traversing the tree (see, for example, step 210 in FIG. 2) to execute the graphics computation on the selected target hardware platform (see, for example, steps 212 and 214 in FIG. 2), Peercy does not expressly disclose the steps of:

(c) storing an application graph that expresses the identity of the stored processing blocks and data connectivity between the stored processing blocks based on the selected target hardware platform; and

(d) traversing the application graph at run time, including executing appropriate processing blocks on the selected target hardware platform.

However, Pavan further discloses storing an application graph (see, for example, user program 400 in FIG. 4) that represents the identity of the stored processing blocks and the data connectivity between the blocks (see, for example, column 5, lines 8-14, and column 6, lines 18-26). Pavan further discloses that the application graph is traversed and translated at runtime to execute appropriate processing blocks on a selected target hardware platform (see, for example, column 8, line 51 to column 9, line 9). The traversal is performed at runtime when the application is submitted for execution (see, for example, column 12, lines 1-13). Program fragments are automatically and transparently created and distributed for execution over a network based on the application graph (see, for example, column 11, lines 50-63).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy with an application graph, such as taught by Pavan, so as to automatically and transparently create and execute a distributed program for the graphics computation of Peercy (see, for example, Pavan, column 11, lines 50-63).

Although Pavan discloses modifying the application graph at runtime (see, for example, column 8, lines 60-67, which shows inserting blocks, and column 11, lines 2-12 and 20-32, which shows modifying the connections between the blocks) when the application is submitted for execution (see, for example, column 12, lines 1-13), Peercy in view of Pavan does not expressly disclose the step of:

modifying the application graph as a result of executing at least one of the appropriate processing blocks.

However, Contreras discloses an application graph (see, for example, FIG. 3) and an engine that traverses the application graph at runtime (see, for example, column 5, lines 13-16).

The application graph includes nodes or processing blocks (see, for example, column 4, lines 27-36) and edges that represent data flow connections (see, for example, column 4, lines 37-44). Contreras further discloses a database or dictionary of nodes (see, for example, column 4, lines 9-11). Contreras expressly discloses modifying the connections among nodes (see, for example, column 5, lines 25-27) and adding nodes to the application graph dynamically at runtime (see, for example, column 5, lines 37-39). The execution of a node or processing block can cause such changes to the application graph, so as to respond to events dynamically without user intervention (see, for example, column 4, lines 45-52 and column 5, lines 55-59).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy and Pavan with the execution features taught by Contreras, so as to respond to events dynamically without user intervention (see, for example, Contreras, column 4, lines 45-52).

With respect to claim 4 (original), Peercy further discloses the limitation wherein the content comprises game content (see, for example, column 1, lines 16-23, which shows developing graphics applications for the entertainment industry, and column 11, lines 1-2, which further shows using video game cartridges), and the multiple hardware platforms include at least one of a game console platform and a personal computer platform (see, for example, FIG. 5, which shows a computer platform, and column 11, lines 1-2, which shows a game device or console platform).

With respect to claim 5 (currently amended), Peercy discloses a game development and run time system (see, for example, the abstract and FIG. 1), comprising a graphical application platform that enables a game application to run on any of multiple hardware platforms (see, for example, column 6, lines 2-20, which shows a graphical application platform that enables an application to run on any of multiple hardware platforms, and column 1, lines 16-23, which further shows developing graphics applications for the entertainment industry).

Although Peercy discloses using an abstract representation to implement standard graphics features and capabilities of a hardware platform (see, for example, column 5, lines 35-45), Peercy does not expressly disclose:

(a) an application real time kernel,

(b) a plurality of standard features implemented as executable blocks of logic, and

(c) connections between said blocks that implement data flow between said blocks such that capabilities of the game application and any of the multiple hardware platforms can be implemented modularly by adding additional corresponding blocks and connections,

wherein the application real time kernel can modify the connections between said blocks at run time.

However, Pavan discloses a kernel for the simultaneous, real-time control of applications having multiple inputs and outputs (see, for example, column 2, lines 3-14). Pavan further discloses a plurality of standard features implemented as blocks (see, for example, column 4, lines 9-18), and connections between the blocks that implement data flow (see, for example, column 5, lines 8-14). Blocks and connections are added modularly to implement the capabilities of the application (see, for example, column 4, lines 28-37) and the hardware platform (see, for example, column 6, lines 45-63). Pavan further discloses that the blocks and connections are modular (see, for example, column 4, lines 19-26), and encapsulate functions performed by hardware and functions that are otherwise hidden from the user (see, for example, column 4, lines 9-18).

Pavan further discloses that blocks are inserted (see, for example, column 8, lines 60-67) and that the connections are modified at runtime (see, for example, column 11, lines 2-12 and 20-32) when the application is submitted for execution (see, for example, column 12, lines 1-13), so as to automatically and transparently create and distribute program fragments for execution over a network (see, for example, column 11, lines 50-63).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Peercy with an application real time kernel, blocks and connections, such as taught by Pavan, so as to encapsulate and provide access to functions performed by hardware and functions that are otherwise hidden from the user (see, for example, Pavan, column 4, lines 9-18), and to automatically and transparently create and execute a distributed program for the graphics features of Peercy that can run in real-time (see, for example, Pavan, column 11, lines 50-63).

Peercy in view of Pavan does not expressly disclose that the limitation above, wherein the application real time kernel can modify the connections between said blocks at run time, happens as a result of executing at least one of said blocks.

However, Contreras discloses an application graph (see, for example, FIG. 3) and an engine that traverses the application graph at runtime (see, for example, column 5, lines 13-16). The application graph includes nodes or processing blocks (see, for example, column 4, lines 27-36) and edges that represent data flow connections (see, for example, column 4, lines 37-44). Contreras further discloses a database or dictionary of nodes (see, for example, column 4, lines

9-11). Contreras expressly discloses modifying the connections among nodes (see, for example, column 5, lines 25-27) and adding nodes to the application graph dynamically at runtime (see, for example, column 5, lines 37-39). The execution of a node or processing block can cause such changes to the application graph, so as to respond to events dynamically without user intervention (see, for example, column 4, lines 45-52 and column 5, lines 55-59).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Peercy and Pavan with the execution features taught by Contreras, so as to respond to events dynamically without user intervention (see, for example, Contreras, column 4, lines 45-52).

With respect to claim 6 (original), although Peercy discloses that the application can run on a target hardware platform (see, for example, column 6, lines 2-9), and further discloses a tree or application graph (see, for example, step 208 in FIG. 2, FIG. 3 and column 4, lines 14-16), Peercy does not expressly disclose an object definition tool that enables a developer to define an application graph.

However, Pavan discloses an object definition tool that enables a user to define an application graph (see, for example, column 6, lines 18-26), so as to simplify the programming for the user (see, for example, column 7, lines 37-49).

It would have been obvious to one of ordinary skill in the art at the time the invention was made supplement the system of Peercy with an object definition tool, such as taught by Pavan, so as to simplify the programming for the user (see, for example, Pavan, column 7, lines 37-49).

With respect to claim 7 (original), Pavan further discloses the limitation wherein said object definition tool further enables a developer to define objects, object elements, and connections (see, for example, column 5, line 25 to column 6, lines 8, which shows that the user defines objects, elements of the objects, and connections between the objects).

With respect to claim 8 (currently amended), the limitations recited in the claim are analogous to those of claim 5 (see the rejection of claim 5 above).

With respect to claim 10 (original), Pavan further discloses the limitation wherein said additional blocks implement additional features, said additional features comprising market oriented features (see, for example, column 4, lines 9-18, which shows that the blocks implement functions or features, and column 1, line 56 to column 2, line 2, which further shows market-oriented features).

With respect to claim 11 (original), Pavan further discloses the limitation wherein said additional blocks implement additional features, said additional features comprising application specific features (see, for example, column 4, line 48 to column 5, line 7, which shows that the blocks implement application-specific features).

With respect to claim 16 (currently amended), the limitations recited in the claim are analogous to those of claim 1 (see the rejection of claim 1 above).

With respect to claim 17 (previously presented), the limitations recited in the claim are analogous to those of claim 2 (see the rejection of claim 2 above).

With respect to claim 18 (currently amended), the limitations recited in the claim are analogous to those of claim 3 (see the rejection of claim 3 above).

With respect to claim 19 (previously presented), the limitations recited in the claim are analogous to those of claim 4 (see the rejection of claim 4 above).

7. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Peercy in view of Pavan in view of Contreras, as applied to claim 8 above, and further in view of U.S. Patent No. 6,584,489 to Jones et al. (art of record, "Jones").

With respect to claim 9 (original), although Pavan discloses resource scheduling and other kernel services (see, for example, column 6, lines 45-63), Peercy in view of Pavan in view of Contreras does not expressly disclose the limitation wherein said ARK comprises logic that invokes blocks according to a schedule listing the blocks to be executed in each of at least one ARK thread running on at least one central processing unit, dynamically loads and unloads blocks, monitors block execution, and facilitates thread management, memory sharing, mutual exclusion, and synchronization.

However, Jones discloses invoking blocks according to a schedule for one or more threads on one or more processors (see, for example, column 6, lines 9-26). Jones further discloses monitoring execution to dynamically load and unload resources or blocks (see, for example, column 14, lines 41-48). Jones further discloses managing threads (see, for example, column 18, lines 18-39) and resources such as shared memory (see, for example, column 5, lines 16-28). Jones further discloses mutual exclusion and synchronization (see, for example, column 25, lines 41-47). The scheduling and management features of Jones provide dynamic and adaptable support for resources in distributed, real-time applications (see, for example, column 4, lines 57-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Peercy, Pavan and Contreras with scheduling and management features, such as taught by Jones, so as to provide dynamic and adaptable support for resources in the distributed, real-time graphics application of Peercy, Pavan and Contreras (see, for example, Jones, column 4, lines 57-67).

8. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Peercy in view of Pavan in view of Contreras, as applied to claim 8 above, and further in view of U.S. Patent No. 5,857,106 to Barbour et al. (art of record, "Barbour").

With respect to claim 12 (original), although Pavan further discloses the limitation wherein said standard and additional blocks are organized into components (see, for example, column 4, lines 19-26, which shows primitive or standard blocks and composite or additional blocks, and column 4, lines 2-5, which shows organizing the blocks into objects or components), Peercy in view of Pavan in view of Contreras does not expressly disclose the limitation wherein each component comprises blocks representing alternative implementations of a feature.

However, Barbour discloses libraries or components comprised of modules or blocks that represent alternative implementations of features for different hardware architectures, so as to provide optimized performance for a selected hardware platform (see, for example, column 1, lines 55-65 and column 2, lines 53-61).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Peercy, Pavan and Contreras with alternative implementations, such as taught by Barbour, so as to optimize performance for a selected hardware platform (see, for example, Barbour, column 1, lines 55-65 and column 2, lines 53-61).

9. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Peercy in view of Pavan in view of in view of Contreras in view of Barbour, as applied to claim 12 above, and further in view of Jones.

With respect to claim 13 (original), Barbour further discloses the limitation wherein each of said alternative implementations comprises:

(a) blocks corresponding to said alternative implementation (see, for example, column 1, lines 55-65, and column 2, lines 53-61).

Although Barbour discloses identifying the processor and architecture (see, for example, column 3, lines 29-39), Peercy in view of Pavan in view of Contreras in view of Barbour does not expressly disclose the limitation wherein each of said alternative implementations comprises:

(b) identification of resources needed by said alternative implementation; and

(c) identification of resources provided by said alternative implementation.

However, Jones discloses identifying resources that are needed (see, for example, column 9, lines 38-61) and resources that are provided (see, for example, column 9, lines 8-14), so as to provide dynamic and adaptable resource management for distributed, real-time applications (see, for example, column 4, lines 57-67).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Peercy, Pavan, Contreras and Barbour with resource identification, such as taught by Jones, so as to provide dynamic and adaptable resource management for the distributed, real-time graphics application of Peercy, Pavan, Contreras and Barbour (see, for example, Jones, column 4, lines 57-67).

10. Claims 14 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Peercy in view of Barbour in view of Contreras.

With respect to claim 14 (currently amended), Peercy discloses a method of executing a feature for a graphics application with respect to a predefined hardware platform (see, for example, the abstract and FIG. 1).

Although Peercy discloses targeting a hardware platform (see, for example, column 6, lines 2-9), Peercy does not expressly disclose:

(a) selecting from among a set of alternative implementations of a feature.

However, Barbour discloses selecting an implementation of a feature (see, for example, column 3, lines 29-39) from among a set of modules or blocks that correspond to alternative implementations (see, for example, column 2, lines 29-39), so as to provide optimized performance for a selected hardware platform (see, for example, column 1, lines 55-65, and column 2, lines 53-61).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy with alternative implementations, such as taught by Barbour, so as to optimize performance for the targeted hardware platform.

Peercy further discloses:

(b) mapping at least one block, corresponding to the selected implementation, to a phase of execution (see, for example, steps 206 and 208 in FIG. 2 and column 5, lines 35-45, which shows mapping an abstract expression or block to an intermediate tree representation, and FIG. 3, which shows that the tree comprises phases of execution);

(c) mapping a phase of execution to a stage of execution (see, for example, step 210 in FIG. 2 and column 5, lines 46-50, which shows processing the tree to map the phases of execution to primitive commands or stages of execution for the underlying API, and FIG. 4, which shows the tree after such mapping);

(d) creating a block execution order list corresponding to the stage of execution (see, for example, step 212 in FIG. 2 and column 5, lines 46-61, which shows creating a sequence or an order list of primitive commands for execution);

(e) submitting the stage of execution to an application real time kernel for management of execution of the stage (see, for example, step 214 in FIG. 2 and column 5, lines 54-56, which shows submitting the sequence for execution).

Peercy in view of Barbour does not expressly disclose the limitation of step (e) wherein the execution of the stage may cause an additional feature to be executed following the steps of (a)-(e).

However, Contreras discloses an application graph (see, for example, FIG. 3) and an engine that traverses the application graph at runtime (see, for example, column 5, lines 13-16). The application graph includes nodes or blocks that are mapped to stages of execution in a finite state machine (see, for example, column 4, lines 27-36). Contreras further discloses that each node or block may itself comprise a finite state machine that is submitted for execution (see, for example, column 5, lines 48-54). The execution of a finite state machine may cause the addition of another node or block (see, for example, column 5, lines 55-59) dynamically at runtime (see, for example, column 5, lines 25-27), which is necessarily executed following the same process, so as to respond to events dynamically without user intervention (see, for example, column 4, lines 45-52).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Peercy and Barbour with the execution features taught by Contreras, so as to respond to events dynamically without user intervention (see, for example, Contreras, column 4, lines 45-52).

With respect to claim 15 (original), Peercy further discloses the limitation wherein said step (a) comprises a negotiation process in which resource requirements of each alternative implementation are considered, along with the costs and benefits of variations in such resource requirements, thereby allowing selection of an implementation (see, for example, column 5, lines 46-61, which shows considering the costs of an implementation in terms of execution time and resource requirements and selecting an efficient sequence).

Allowable Subject Matter

11. Claims 20 (currently amended) and 21-23 (previously presented) are allowed. The following is a statement of reasons for the indication of allowable subject matter:

The prior art of record does not expressly disclose executing an application graph, wherein the application graph is dynamically modified, and wherein said executing step includes

(a) traversing the application graph, (b) executing processing blocks of the application graph, wherein executing the processing blocks specifies a necessary feature, (c) selecting a component from a dictionary based on the specified necessary feature, (d) selecting an implementation from the selected component based on the hardware configuration, and (e) modifying the application graph dynamically at runtime by inserting the set of processing blocks of the selected implementation into the graph, in such a manner and combination as recited in independent claim 20. Dependent claims 21-23 further limit the method of claim 20.

Conclusion

12. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER